Strategyproof Computing: Systems Infrastructures for Self-Interested Parties

Strategyproof Computing: Systems Infrastructures for Self-Interested Parties

Chaki Ng, David C. Parkes, and Margo Seltzer
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
[chaki,parkes,margo]@eecs.harvard.edu

ABSTRACT

The widespread deployment of high-speed internet access is ushering in a new era of distributed computing, in which parties both contribute to a global pool of shared resources and access the pooled resources to support their own computing needs. We argue that system designers must explicitly address the self-interest of individual parties if these next-generation computing systems are to flourish. We propose *strategyproof computing*, a vision for an open computing infrastructure in which resource allocation and negotiation schemes are incentive-compatible, and individual parties can treat other resources as their own. In this paper we outline key guiding principles for the vision of strategyproof computing, define the strategyproof computing paradigm, and lay out a systems-related research agenda.

1. INTRODUCTION

The widespread deployment of high-speed Internet access, including the rapid build-out of WiFi capability, is leading to massively distributed computing systems, including peer-to-peer systems, pervasive systems [27], computational grids [4], and overlays on the Internet (such as distributed caching). More than being physically distributed, these systems are also designed, owned, and used by multiple autonomous and self-interested parties. As such, these systems adopt the properties of both a computational system and an economic system.

Two points cannot be understated. First, these systems are truly *open*, with innovation and competition and little central control beyond agreement on fundamental communication protocols such as those within the current Internet. This open nature is vital for the continued design and deployment of new and rich capabilities, and must be explicitly recognized and embraced.[1] Second, the participants in these systems are truly self-interested (at least to the same extent that people and businesses in human societies are self-interested) and heterogeneous in both their capabilities and their goals.

Today, most distributed systems and protocols assume that parties are either *obedient*, and do not deviate from system protocols, or *malicious* or perhaps simply *faulty*. As such, these systems fail to address the challenges that are introduced when *rational* behavior (such as a deliberate misstating of requirements or capabilities to improve the utility of an outcome for an individual user) is the norm in systems [3, 22]. Looking ahead, we believe that it is essential to embrace incentives in the design of systems that sustain useful coordinated behaviors in the long-term. We introduce the paradigm of *strategyproof computing*, a vision that explicitly recognizes and embraces the self-interest and incentives of individual parties in distributed systems.

Self-interest manifests itself in many compelling future scenarios for next-generation computing. As an example, consider scenarios in pervasive computing [20]. These scenarios typically depict worlds where computing elements collaborate effortlessly behind-the-scene, in smart environments. Imagine a mobile user travelling across towns. Her devices constantly seek services, such as short term data storage or computing resources (to host voice reconition or audio playback capabilities), downloadable maps (complete with local advertisements), or realtime news and traffic conditions. For each service there will be tens or hundreds of competing providers. Likewise, for each provider there will be significant resource contention and profit decisions. This is a dynamic and multiparty negotiation problem, that presents many realistic questions that are not addressed with models that assume cooperation or occasional malicious or faulty behavior.

Strategyproof computing considers the problem of designing systems with useful long-term equilibrium behavior, when users (or their computational agents) have presumably figured out how to "play the game" to their advantage. We build on two intellectual threads: *market-based approaches* to resource allocation and the economic theory of *mechanism design*. In particular, we advocate that distributed systems be populated with strategyproof mechanisms but without requiring one specific mechanism.

Strategyproof computing is an *infrastructure effort* that seeks to enable multiple participants— service providers, resource providers and intermediaries —to describe and deploy new mechanisms, centered around their services and resources, in an open playing field. A critical role of the infrastructure is to validate the incentive properties of mechanisms. Without this validation mechanisms cannot commit to implement particular rules and the useful simplifying effect that mechanisms can have on the strategies facing participants can quickly unravel.

In a companion paper [14], we lay out some of the AI-related challenges. Here, we focus on systems-related challenges in making the vision of strategyproof computing a reality. We also seek to motivate this "incentives-first" agenda and to present a broad-brush view of what we see as the central components of a strategyproof-computing architecture.

[1] Indeed, the original Internet design was built around an end-to-end principle, that explicitly left the "smarts" to the edges of the network to allow for the development and deployment of rich and unanticipated applications [18].

2. STRATEGYPROOF COMPUTING

In this section, we begin with some guiding principles that direct the paradigm of strategyproof computing (SPC). Continuing, we describe the main ingredients— local strategyproof mechanisms and an open market for mechanisms —and outline three core architectural components.

In distinguishing SPC from the many earlier proposals for economically-motivated mechanisms for decentralized resource allocation in systems (e.g. [2, 26, 6, 16, 7, 23]) we emphasize that this is an infrastructure effort. We contend that no single mechanism can possibly be appropriate for all requirements in a heterogeneous system. Instead, we seek to provide the equivalent of the dialtone for the development and deployment of computing services that explicitly handle incentive considerations.

2.1 Guiding Principles

[A1 incentives-first] The self-interest of participants in distributed systems should be explicitly addressed by system designers, much as we currently address issues of fault-tolerance and security.

[A2 utility-based] Resource allocation decisions should be made in a framework that considers the *utility* of users for different outcomes as the basis for arbitration. Moreover, this utility should be measured in terms of a common currency, to allow for an explicit tradeoff between multiple users and across different components of a system.

[A3 simple] As designers we should seek to simplify the decisions facing participants in distributed multi-agent systems, such as the optimal statements to make about the local capabilities (e.g. available storage space, speed of Internet connection, etc.), or the optimal statements to make about the utility of a party for different levels of resources (e.g. bandwidth connectivity).

[A4 open] Systems should be *open* to allow for innovation and competition in the design of new services. This principle applies to the methods of resource allocation and arbitration, just as it applies, for example, to the design of web servers and the design of distributed file caches. A successful open system requires a *lightweight* infrastructure in which we only introduce universal and minimal components, to enable the deployment of services without preventing the future deployment of unanticipated applications.

[A5 decentralized] The control structure in distributed computing systems must be decentralized, both to respect the autonomy of the nodes that own the property rights to resources, but also for reasons of computational scale and timeliness of information.

2.2 Economic Foundations

Given that self-interest exists in distribtued systems, it is natural to turn to economic theory to better understand how to design distributed systems. But, *which* theory?

In what follows, we lay out the two dueling economic foundations for SPC. On one hand, we advocate adopting certain components from the theory of economic mechanism design (e.g. [9, 5]). Mechanism design (MD) explicitly models each participant as a game-theoretic agent and seeks to design optimal incentives to implement desired system-wide outcomes. On the other hand, we take a somewhat relaxed view of MD and propose an open *market for mechanisms* in which multiple competing parties can deploy mechanisms that must only satisfy certain local incentive properties, essentially within the scope of their deployment.

In accepting multiple mechanisms each of limited scope we necessarily move away from the perfect optimality and perfect incentive properties of the pure theory of MD. In particular, there are important questions to address about how participants can best compose services across multiple mechanisms. However, we believe that this relaxation is necessary in the real world setting of distributed computing. Further, we believe that allowing an open system will provide incentives for the development and deployment of mechanisms of an appropriate scope.

In MD, each participant in a system is modeled as a game-theoretic agent, able to compute its best-response within an equilibrium of a system. In the context of distributed systems, this is akin to expecting a user to determine just the right way to state her resource needs to maximize the utility that she receives from participation in the system. With this assumption, MD seeks to design protocols, or "rules of the game", to achieve particular desiderata such as allocative efficiency, fairness, or maximal throughput. MD extends to very general problems, and there are a number of celebrated mechanisms addressing various requirements (see Jackson [5] for an accessible introduction to some of these mechanisms.)

One particularly compelling notion from MD is that of *strategyproof* mechanisms. These are mechanisms in which a self-interested party can maximize her own utility through straightforward truth-revelation about requirements and capabilities, *whatever the strategies and behaviors of other parties*. Technically, truth-revelation is said to be a *dominant strategy*. Participants are thus freed from the complexity of modeling other participants [24, 12]. A common example of strategyproof mechanisms is the second price auction.[2] Thus, strategyproofness nicely addresses the design principles of *incentives-first*, *utility-based* and *simple*.

Yet, MD while beautiful in theory is brittle and unworkable in its purest form. MD advocates that one central mechanism be designed for an entire system, and as such is in direct conflict with the design principles of *open* and *decentralized*. MD also ignores the computational constraints on a solution, and requires that users can all agree on a set of desiderata.[3] In addition, the desirable simplicity that comes from strategyproofness comes at an economic cost, and there are a number of well-known economic limitations of strategyproof mechanisms (e.g. [5]).

Thus, we must relax the goals of MD. Informally, although there has been plenty of interest in developing combinatorial auctions (e.g. [17]), in which heterogeneous auctions are simultaneously offered for sale and participants can submit explicit bids for bundles, we cannot simply build a combinatorial auction for the entire world! Instead, the SPC paradigm recognizes that realistic mechanisms must necessarily have limited scope, both to maintain a reasonable computational and informational scale and because the sphere of influence and control for a single party must necessarily be limited [29]. SPC considers the existence of multiple mechanisms, each of which is locally strategyproof within its scope.

Formally, a ***locally strategyproof mechanism*** $\mathcal{M}$ is one in which truth-revelation is a dominant strategy for any participant that can express her utility for the resources and services within the scope of $\mathcal{M}$ *independently* of the other events in the world, and that chooses to submit requests for those resources *exclusively* to the mechanism. As an example, consider a user with value of \$0.10 to down-

[2] In a second price auction, the bidder with the highest bid wins the auction, and is asked to pay the second highest bid. Bidders should state values truthfully, because the bid price indicates the range of prices a bidder is willing to accept while the actual price paid is determined by the second-highest bid [25].

[3] Parkes [13, chapter 3] provides an extended introduction to issues in computational mechanism design.

load a 1 GB file, irrespective of any other resources she receives (e.g. file space, processing time, etc.). An allocation mechanism is locally strategyproof (LSP) if she can maximize her expected utility by truthfully announcing this file size and value to the mechanism, once she has elected to deal exclusively with that mechanism. Truth-revelation should be her optimal strategy whatever the state of the mechanism and whatever the requests from other participants.

The SPC infrastructure must provide support for multiple users to design and deploy competing LSP mechanisms, in a ***market for mechanisms.*** With this, we achieve the second set of design principles of *open* and *decentralized* systems. Our belief is that an open marketplace will naturally lead to mechanisms with the "right scope" and the "right complexity". This decision represents a tradeoff between providing a large enough scope to sufficiently simplify the game-theoretic decision facing a participant—for example, bringing resources that are *complementary* for a large number of users into the same scope —while maintaining a small enough scope to build computationally reasonable resource allocation mechanisms. The degree to which market forces lead to the emergence of mechanisms with the right scope is an important research question.

2.3 Basic Components

We define three core components for all strategyproof computing systems (see Figure 1).

2.3.1 LSP mechanisms.

First, resource providers or intermediaries (which make profits by facilitating between providers and buyers) will create LSP mechanisms for admittance into the distributed system.

2.3.2 Public interface.

Second, each LSP mechanism provides a public interface that will facilitate how parties can find mechanisms, and how parties can interact with multiple mechanisms and compose services across mechanisms. Part of this interface will make claims about incentive properties, and about statistical properties on the utility of participants that hit a mechanism. The interface must specify:

1) a *service request language*, which is the language to describe how others can query the service. A simple example of what such language should include at a minimum: a service request message that allows a party to express her value and the resource she requests; and a service response message for the party to accept or reject.

2) restrictions on the *preferences* of the parties that are submitting requests, for which the incentive properties of the mechanism hold. This is important, because strategyproofness will often depend on the particular type of goal that a user is seeking to accomplish.

3) *utility statistics*, to provide information on the average utility that parties have realized in the mechanism in the past.

2.3.3 Validation

Third, the infrastructure will assume the key role of providing validation of the incentive and statistical properties of LSP mechanisms. Suppose that a mechanism claims it is LSP and is admitted into the system. How can we prevent it from lying? This desired prevention is the reason that validation schemes must be provided, by the infrastructure, for strategyproof systems to function well.

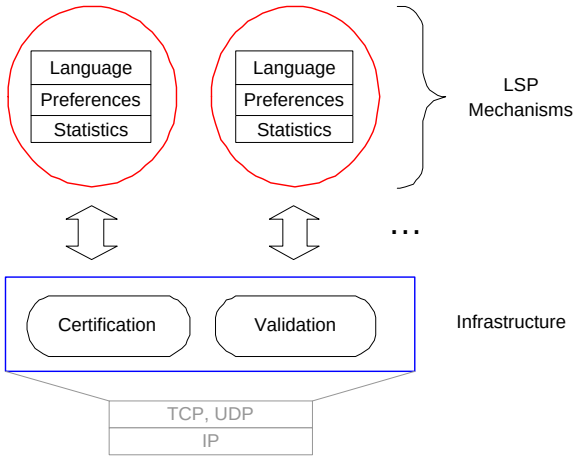

Figure 1: Components for Strategyproof Computing Systems

Indeed, this is the key role of a shared strategyproof computing infrastructure. The truth-revealing equilibrium of well-designed mechanisms can quickly unravel without the ability to commit to a particular set of rules.[4]

Fortunately we believe that this validation can be adequately provided without requiring the direct checking of the rules of mechanisms. Rather, we propose to simply check input and output streams for correct behavior. This provides for only a weak form of violation, i.e. that there is no evidence to support the failure of incentive properties. However, it seems useful in that it checks for failures around the operating point of mechanisms because it is based on actual inputs and outputs.

The infrastructure will assign a certificate to a newly accepted and LSP-claimed mechanism, that can then present the certificate to other parties so they can interact truthfully with confidence. The infrastructure must validate an LSP mechanism as frequently as necessary to maintain the integrity of the certificate.

We propose to check the claimed incentive properties by inspection of the requests made to a mechanism, and its responses. In one approach, we can use *active monitoring*, with policing agents (that belong to the infrastructure) used to poll LSPs and check for deviations from claimed incentive properties. A police agent must be able to masquerade as a user and observe LSP behaviors. Second, we can use *passive monitoring*, in which a subset of requests and responses are monitored (for example via requirements that data be stored for audit purposes). We imagine similar monitoring techniques to check the statistical claims of LSPs.

3. VALIDATION EXPERIMENTS

The validation schemes are crucial to the robustness of an SPC-based system. The novelty of our proposal is that validation will be performed by the infrastructure without any knowledge of the auction algorithms. Indeed, the inputs for validation will be the bids submitted to LSPs, along with the resulting allocation and payments. The research challenge is to understand how to do this efficiently, in terms of computational and space complexity. For now

[4]Consider a Vickrey (second-price sealed-bid) auction in which the auctioneer cannot commit to clearing the auction at the second highest price. Such an auction would degenerate into a first-price auction.

we settle for a limited proof of concept. In this section, we present the results of an experiment to measure the typical number of auctions required to detect a non-LSP mechanism in a simple setting.

We consider a multi-unit allocation problem, with M identical items and single-minded agents that require a particular bundle of items. We model the valuation of agent i by choosing the number of units $k_i \sim U(1, 10)$, with value $v_i \sim U(1, k_i)$. We considered settings with $N = 20$ agents and choose M from $\{20, 30, 40\}$. To keep things simple we simulate truthful agents. However, the validation scheme does not require truthful agents, all that is required is a stream of inputs and outputs into the mechanism to enable the collection of counterfactual information to demonstrate failure of incentive properties. In future work we will study the effect of equilibrium behavior on the ability to quickly detect the failure of LSP properties.

We consider a simple first-price greedy auction, which is a non-LSP mechanism. Agent i can submit a bid (x_i, p_i) pair, for x_i units at total price less than or equal to p_i. The auction sorts the bids in descending-order of unit-price p_i/x_i, and allocates items to a bid while there are still items left to allocate. Winning bids pay their bid price. This auction is not strategyproof, because a winning bidder can reduce her payment and increase her utility by bidding the minimal unit price at which her bid is still successful.

The utility of agent i with value (k_i, v_i) given a bid (x_i, p_i) is simply stated as:

$$u_i(k_i, v_i; x_i, p_i) = \begin{cases} v_i - p_i & \text{, if successful and } x_i \geq k_i \\ 0 & \text{, otherwise.} \end{cases}$$

Let $\mathcal{B}$ denote the set of bids submitted to an auction. In validation, we take bids $(x_i, p_i) \in \mathcal{B}$, and check whether any agent could have increased her *stated* utility by submitting the bid of any other agent. If a mechanism is LSP, then it is *necessary* that

$$u(x_i, p_i; x_i, p_i) \geq u(x_i, p_i; x_j, p_j) \quad (1)$$

for all pairs of bids $i, j \in \mathcal{B}$. The LHS is the utility agent i receives from her bid with respect to her stated valuation. The RHS is the utility agent i would have received from the bid of agent j, again with respect to her stated valuation. Thus, to check for the failure of LSP it is sufficient to find a *single pair* of bids for which condition (1) fails. Informally, it will fail when a losing bidder could have won and still received positive utility from submitting some other bid, or when a winning bidder could have still won her desired number of items but for a lower price by submitting some other bid.

Figure 2 depicts the result of 100 auction trials for each of the different number of items in the auction ($M = \{20, 30, 40\}$). We plot the distribution over the number of auctions required to detect a non-LSP auction. Overall, all three curves are encouraging in that detection is made relatively early— on average well before the tenth auction trial. Most detections happen during one of the first four auctions.

Validation becomes more effective with higher numbers of items because there are more winners, and thus there is more useful counterfactual information with which to detect incentive violations. Notice that losing bids can never provide useful information on the RHS of expression (1).

4. SYSTEMS RESEARCH CHALLENGES

The vision of strategyproof computing is ambitious and presents a large number of research challenges. We lay out some of the challenges in this section.

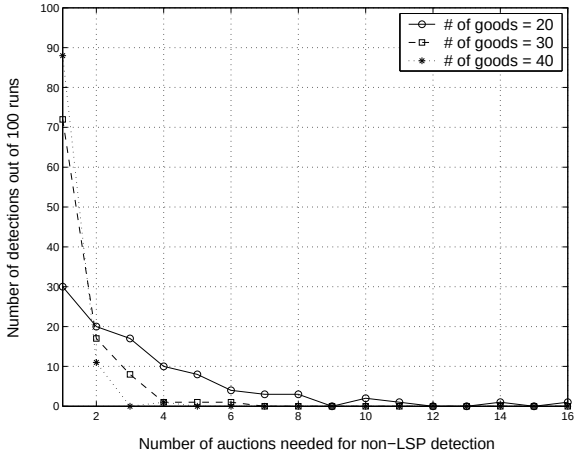

Figure 2: Validation Simulations for a Simple First Price Auction.

A companion paper lays out some of the *AI-related* challenges [14]. These AI challenges include those of *computational mechanism design*, the design of *description languages*, *inference methods* for verification, methods for *automated composition*, and *machine learning* methods to identify a good scope for new mechanisms and to automatically tune existing mechanisms.

4.1 LSP Mechanisms

A basic research goal is to establish the core validity of local strategyproofness in the context of systems settings. This paper is not the place for an extended discussion of the progress that has been made in recent years on the characterization of strategyproof mechanisms (see [14] for some discussion). Simple yet strategyproof mechanisms are often *price-based* (e.g. [8, 11]) or based around generalizations of the philosophy enshrined in Vickrey's auction (e.g. [28, 10]). However, there is still plenty to do. An important question to ask in the context of SPC, is *how expressive are strategyproof mechanisms*, given that we allow them to be limited in scope?

One part of establishing the core validity will be to provide *development tools* that will help developers to program and debug mechanisms from the ground up. It will also be useful to have prototypes of how all these things fit together as soon as possible.

4.2 Validation and Verification

The active and passive monitoring techniques suggested for validation will be most effective only if the responsible parties cooperate: the "police" in active monitoring must be fully trusted by the whole system and must not be suspects of fraud; and passive monitoring requires every party in the system to provide true information in a timely and accurate fashion. These imply serious research efforts in understanding and building trust and reputation techniques.

The computational and economic efficiency of monitoring must be measured and quantified. Non-LSP mechanisms must be detected early and frequently enough. There will be many types of mechanisms and many of them are one-time participants. We must have ways to stop such short-lived mechanisms from hurting other parties. Would an evaluation period, during which a new mecha-

nism will be validated before deployment, be necessary or be limiting with certain services? Furthermore, we need to determine how much information should the audit trails provide for good validation.

For the purpose of the current discussion we have assumed a single infrastructure operator, who will assume responsibility for validation. For fully-decentralized systems such as the Internet and peer-to-peer systems, how should this be adapted? Do we need something like third party organizations such as certificate authories to provide LSP certificates and validation? Can groups of parties assume an *ad-hoc* role in performing decentralized validations?

4.3 Scalability and Decentralization

Traditional MD is completely centralized; a single node collects information from all parties and computes an outcome. Even the limited decentralization of LSP mechanisms will not always be practical in our setting, so we need to target strategies and algorithms for decentralized mechanism design [3]. The new challenge is that the mechanism is implemented by the same nodes that stand to gain from the resource-allocation decision, which means that it must be rational for these nodes to choose to implement the computation and message-passing required to deploy the mechanism. We will develop fundamental building blocks, including redundancy, partitioning, and dynamic selection, to address these challenges [22].

4.4 Composition

As we pointed out, a group of LSP mechanisms that a party interacts with is not necessarily strategyproof - the party may behave strategically to obtain the highest possible utility from one of the mechanisms. How can we scale up with distinct LSP mechanisms? We need to be able to *discover and compose* LSP mechanisms to form larger-scope LSP mechanism sets. We can treat the individual mechanisms as the building block for these sets, which will be the normal scope an average party would deal with (in other words we do not forsee parties being satisfied dealing with just one mechanism for each resource).

One appealing methodology is to develop mechanisms that provide *real options* to users, rather than outcomes from which a user cannot decommit. A party that wants to compare among several LSP mechanisms may now first iteratively interact with each mechanism truthfully to obtain an option. She could then simply compare the options and decide which option to exercise.

4.5 Measurements

The negative effects of self-interest within systems must be clearly quantified. System designers are typically aware of self-interest behaviors, but there is no absolute measurement of how much they affect the systems. Past attempts have pointed out ways in which self-interested parties can alter a system's topology [19], but there does not seem to be much urgency in addressing incentive issues. The main difficultly is what to measure. For some systems, the crucial factors may be how many parties are or are not contributing resources. For others, it maybe how much efficiency the overall system achieves in utility terms [21, 1]. We note that our infrastructure proposal exposes utility information of different parties.

4.6 Deployment Issues

Deployment is the next large hurdle. Until real LSP mechanisms join a particular system, the parties in the system may not observe direct benefits of strategyproof computing and hence may leave. Likewise, there will be less incentive for any one to develop LSP mechanisms.

How can we begin to deploy strategyproof systems? If we aim for the Internet, would that be too unrealistic? The Internet is where rapid adoption occurs, when one does things 'right.' On the other hand, we can use start-up systems such as PlanetLab [15] as a testbed. What if we take the hybrid approach and target up-and-coming systems such as specific grid systems? These have a larger user base than start-up systems, and scientists seem to be getting their experiments done.

Strategyproof computing will affect the parties in a system in both positive and perhaps negative ways. It is important that we identify these effects early on to determine the incentives for particular parties to participate. Overall, we see that users have a lot to gain in strategyproof systems, as they can interact truthfully and minimize interactions to get their resources while still can extract desirable utilities. Overall, although resource providers have costs to develop LSP mechanisms, these investments will be paid off in the long run with users that appreciate the transparency and simplicity provided by the SPC paradigm. It is also important to note that running LSP mechanisms does not imply the inability to extract revenue. However, in the short term the sophisticated users and resource providers in current systems may be hesitant to join strategyproof systems.

5. CONCLUSION

Strategyproof computing is an incentives-first agenda that proposes research into the design of an appropriate infrastructure to provide the ability to embrace incentives and allow the build-out of mechanisms to handle self-interest in distributed systems. We believe that it is important to recognize the existence of incentive issues in next-generation distributed systems, such as peer-to-peer systems, and to design protocols to *simplify* the strategic choices facing users.

Any new vision takes years or perhaps even decades to realize, if at all. For example, we are only now moving close to the vision of pervasive computing [27], many years after it was first articulated. Nonetheless, for strategyproof computing, we cannot afford to wait too long. Systems are getting more sophisticated, and tensions among parties are only going to get higher. Instead of spending years trying to build the ideal strategyproof computing blueprint, we hope to begin to quantify strategyproof computing costs and benefits, provide real development methodology and tools, and transform some real existing distributed systems into (even if not perfectly) strategyproof ones.

Acknowledgements

The authors gratefully acknowledge recent conversations with John Wroclawski and Michael Wellman. This work is supported in part by NSF grant IIS-0238147.

6. REFERENCES

[1] R. Braynard, D. Kostic, A. Rodriguez, J. Chase, and A. Vahdat. Opus: An overlay peer utility service. In *Proc. 5th Int. Conf. On Open Architectures and Network Programming (OPENARCH)*, 2002.

[2] S. H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[3] J. Feigenbaum and S. Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, 2002.

[4] I. T. Foster and C. Kesselman. Computational grids. In *VECPAR*, pages 3–37, 2000.

[5] M. O. Jackson. Mechanism theory. In *The Encyclopedia of Life Support Systems*. EOLSS Publishers, 2000.

[6] J. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. on Computers*, 38:705–717, 1989.

[7] J. O. Ledyard, C. Noussair, and D. Porter. The allocation of a shared resource within an organization. *Economic Design*, 2:163–192, 1996.

[8] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, September 2002.

[9] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[10] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proc. 18th National Conference on Artificial Intelligence (AAAI-02)*, July 2002.

[11] C. Ng, D. C. Parkes, and M. Seltzer. Virtual Worlds: Fast and Strategyproof Auctions for Dynamic Resource Allocation. In *Fourth ACM Conf. on Electronic Commerce (EC'03)*, 2003. Poster version. Longer version at http://www.eecs.harvard.edu/econcs/virtual.pdf.

[12] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[13] D. C. Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, May 2001. http://www.cis.upenn.edu/~dparkes/diss.html.

[14] D. C. Parkes. Five AI Challenges in Strategyproof Computing. In *Proc. of IJCAI Workshop on AI and Autonomic Computing*, August 10 2003. Available at http://www.eecs.harvard.edu/econcs/ai-spc.pdf.

[15] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A blueprint for introducing disruptive technology into the internet, 2002.

[16] O. Regev and N. Nisan. The POPCORN market-an online market for computational resources. In *Proceedings of the first international conference on Information and computation economies*, pages 148–157. ACM Press, 1998.

[17] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[18] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, pages 277–288, 1984.

[19] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[20] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, pages 10–17, Aug 2001.

[21] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 47–57, 1994.

[22] J. Shneidman and D. C. Parkes. Rationality and self-interest in peer to peer networks. In *2nd Int. Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.

[23] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, and C. Staelin. An economic paradigm for query processing and data migration in Mariposa. In *Proc. 3rd Int. Conf. on Parallel and Distributed Information Systems*, pages 58–67, 1994.

[24] H. R. Varian. Economic mechanism design for computerized agents. In *Proc. USENIX Workshop on Electronic Commerce*, 1995. Minor update, 2000.

[25] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[26] C. A. Waldspurger, T. Hogg, B. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A distributed computational economy. *IEEE Trans. on Software Engineering*, 18:103–117, 1992.

[27] M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–100, Sept 1991.

[28] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271–303, 2001.

[29] M. P. Wellman and P. R. Wurman. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24(3–4):115–125, 1998.